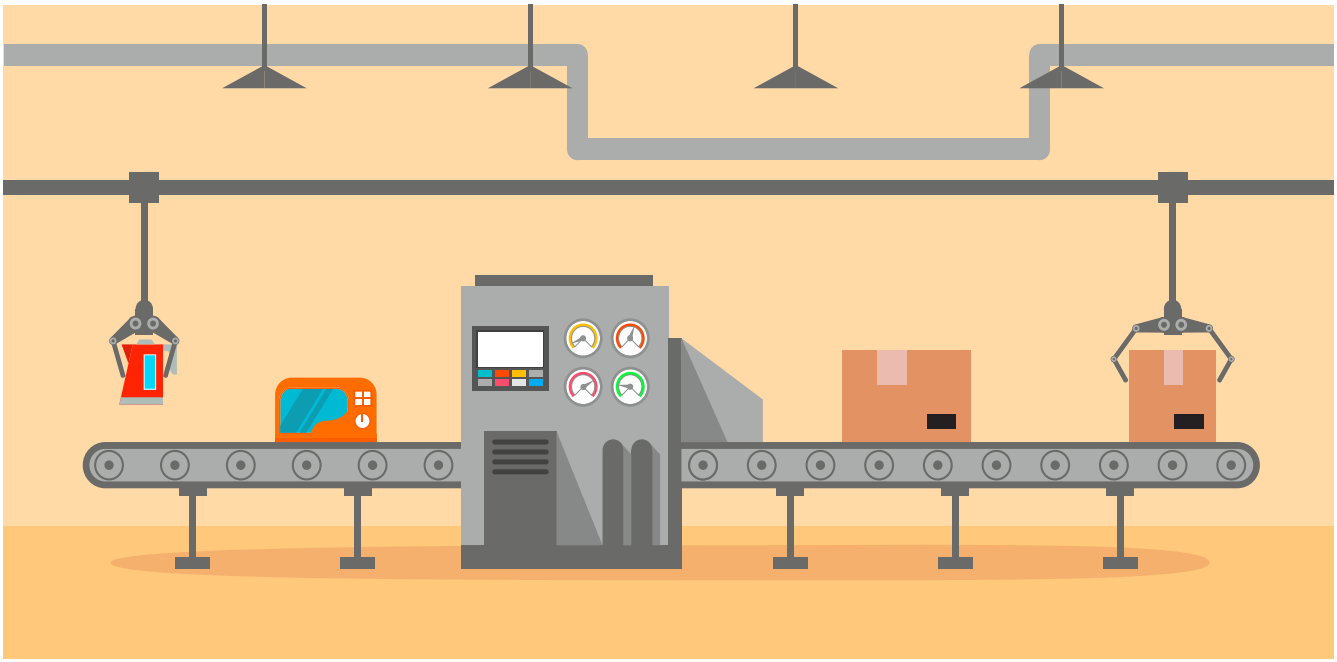
Sonderdruck aus BI-Spektrum 3/2017



Generische Data-Warehouse-Automation

Standardisierung als Erfolgsmodell

Ein Beitrag
von Thomas Weiler und
Annkathrin Wagner

Über Jahrzehnte hat sich in der Praxis eine Basis-Schichtenarchitektur eines Data Warehouse (DWH) herauskristallisiert. Doch DWH-Projekte mit klassischen Werkzeugen zum Beispiel zur Modellierung und Datentransformation verlieren sich häufig in der Definition optimaler Datenstrukturen und der Implementierung massenhafter Datenprozesse. Fehlende Standards, Metadaten und vernachlässigte Dokumentation tun ein Übriges, um für lange Projektlaufzeiten und nicht eingehaltene Budgetvorgaben zu sorgen. Die Data Warehouse Automation (DWA) liefert eine echte Alternative für Entwickler und bietet Unternehmen signifikante Vorteile.

Eine Data-Warehouse-Architektur besteht heute in der Regel aus drei Schichten: Eingangsschicht (Stage), zentraler Datenhaushalt (Spot) als „Single Point of Truth“ und analytischer Datenbestand (Mart). Diese Standardschichten sind hinsichtlich ihrer Modellierung und Persistenz flexibel gestaltbar. Typischerweise findet sich in der Eingangsschicht des Data Warehouse (DWH) eine strukturelle Kopie der Quelldatenstrukturen, die mit zusätzlichen technischen Informationen zum Datenmanagement angereichert ist.

Ähnlich verhält es sich mit dem Spot, der in der Regel ein normalisiertes Modell, eine Ankermodellierung oder ein Data-Vault-Modell aufweist. Marts hingegen werden dimensional nach dem Star- oder Snowflake-Schema modelliert. Wie die einzelnen Schichten ausgestaltet werden, hängt von der konkreten Situation ab, hierzu gibt es grundsätzlich kein Richtig oder Falsch.

DWHs mit perspektivischer Ausrichtung auf eine unternehmensweite Datenintegration sind als Hub&Spoke-Architektur nach Bill Inmon bekannt. Sie favorisieren die vollständige Umsetzung aller Schichten. Dies verursacht, im Gegensatz zur Bus-Architektur nach Ralph Kimball, einen hohen initialen Aufwand, denn hier stehen die analytischen Informationen im Vordergrund. Mittlerweile hat sich durch die technische Weiterentwicklung der Hard- und Software auch eine Mischform etabliert, die Spot und Mart vermengt. Sie bietet zusätzlich für die Datenanalyse eine logische Sicht auf diese Schicht.

Alle drei Architekturen (siehe schematisch in Abbildung 1) haben gemeinsam, dass sie auch die Grundmechanismen der Datenüberführungsprozesse bestimmen, beispielsweise zur Versionierung von Informationen. Zur Umsetzung von „Slowly Changing Dimensions“ existieren für alle Schichten und Modellierungstechniken Best Practices zur technischen Umsetzung. Einer automatisierten Implementierung sollte damit nichts mehr im Wege stehen.

Status quo der DWH-Automatisierung

Um ein DWH schnell und effizient zu implementieren sowie stabil und performant zu betreiben, werden in DWH-Projekten viele Werkzeuge eingesetzt: Modellierungs-, ETL-, Deployment-, Workflow-, Datenqualitäts-, Monitoring- sowie Analysewerkzeuge, teilweise sogar mehrere der gleichen Art parallel. Jedes Werkzeug folgt seiner eigenen Philosophie, benötigt ein eigenes Repository und verfügt über eigene Dokumentationsmöglichkeiten. Jedoch verlieren sich DWH-Projekte trotz all dieser spezialisierten und mächtigen Werkzeuge zu häufig in der Definition optimaler Datenstrukturen und der Implementierung massenhafter Datenprozesse. Es fehlt an übergreifender Standardisierung ebenso wie an konsolidierter Dokumentation. Die Data Warehouse Automation (DWA) liefert hier eine echte Vorgehensalternative.

Die Basis der DWA ist die „Standardisierung" über alle Artefakte, die in einem DWH-Projekt relevant sind. In der Software-Entwicklung ist die Standardisierung unter der Bezeichnung „Design Pattern" seit langer Zeit bekannt. In der DWH-Entwicklung wird sie immer noch stiefmütterlich behandelt. ETL-Werkzeuge bieten heutzutage sogenannte Wizzards an, zum Beispiel für die Versionierung; diese haben aber keinerlei Bezug zum verwendeten Datenmodell. Im Gegenteil, das ETL-Werkzeug bestimmt meist die notwendige Datenstruktur. Modellierungswerkzeugen fehlt es ebenfalls an standardisierten Templates zur Entwicklungsunterstützung. Das Rad wird quasi in jedem DWH-Projekt wieder neu erfunden.

Standardisierung als Basiskonzept

Wie am Versionierungsbeispiel zu sehen ist, stellen sich bei der DWH-Entwicklung immer wieder die gleichen Aufgaben. Was liegt also näher, als diese zu standardisieren und damit automatisierungsfähig zu machen? Durch Standardisierung verliert man sich nicht länger in technischen Fragestellungen, sondern konzentriert sich auf fachliche Anforderungen. Dies verkürzt die Entwicklungszeit und gewährleistet eine hohe Qualität und Performance. Darüber hinaus lassen sich im Rahmen der Standardisierung auch neue Technologien wie zum Beispiel ein Data Lake integrieren.

DWA legt die Nutzung von Entwurfsmustern über alle Kernprozesse bei der Erstellung eines DWH als oberste Prämisse fest und abstrahiert von technischen Umsetzungsdetails. Für ein zentrales DWH-Modell kann der Designer zum Beispiel unterschiedliche „Muster" wie normalisierte Ankermodellierung oder Data Vault wählen. Ist die Wahl getroffen, hängt die technische Umsetzung nur noch von den Attributausprägungen ab und kann vollständig automatisch erfolgen. Dies kann modellgetrieben als Top-down-Ansatz oder datengetrieben als Bottom-up-Ansatz erfolgen [Eck 15].

Konkrete DWH-Entwurfsmuster finden sich auf unterschiedlichen Ebenen eines DWH-Systems. Neben der Schichten-Architektur sind folgende Muster typische Artefakte eines DWH-Projekts:

- **Datenstrukturmuster** zur konzeptionellen und physischen Ablage von Daten
  - **Datenmodellierung:** (De)normalisiert, Ankermodellierung, Actual/History Table, Rekursiv, Data Vault, Dimensional, Star-/Snowflake, konforme Dimensionen
  - **Datenspeicherung:** Relationale/Multidimensionale Datenbank, zeilen-/spaltenbasiert, In-Memory, Cloud, NoSQL, HDFS-basiert (zum Beispiel Hadoop)
- **Datenmanagement-Muster** zur Verarbeitung und Fortschreibung
  - **Mandantentum** zur physischen/logischen Trennung mandantenbezogener Daten
  - **Schlüsselkonzept** zur Verwendung natürlicher Schlüssel (sogenannte Business Keys) versus Surrogaten oder explizitem Schlüssel-Mapping
  - **Versionierung** mit aktuellen Snapshots oder Historie (Slowly Changing Dimensions), unter Umständen über bitemporale Versionierung mit technischer und fachlicher Sichtweise
  - Abbildung einer **Mehrsprachigkeit**
  - **Coding** von objektbezogenen Code-Ausprägungen oder zentralem Code-Mapping bzw. explizitem Stammdatenmanagement
- **Datenintegrationsmuster**
  - **Technologie** zur Datenfortschreibung mittels ETL oder ELT/In-Database
  - **Techniken und Funktionen** zur Integration wie beispielsweise Datenakquisition im Push-, Pull- oder Replizierungsverfahren, logische Transformationen wie Filter, Konformität, Cleansing oder Aggregation sowie Lademechanismen im Append-, Replace- oder Update-Verfahren, Indizierung und Parallelität

THOMAS WEILER ist BI-Experte bei mayato. Er verfügt über die Erfahrung aus zwei Jahrzehnten erfolgreicher Tätigkeit in der Konzeption, Modellierung, Realisierung und Optimierung von Business-Intelligence-Systemen verschiedenster Hersteller.
thomas.weiler@mayato.com

ANNKATHRIN WAGNER erstellt als Team- und Projektleiterin bei mayato Infrastrukturen für Data Warehousing und Big Data, Berichts- und Dashboard-Applikationen sowie Analytics-Lösungen, insbesondere im Bereich Marketing und CRM.
annkathrin.wagner@mayato.com

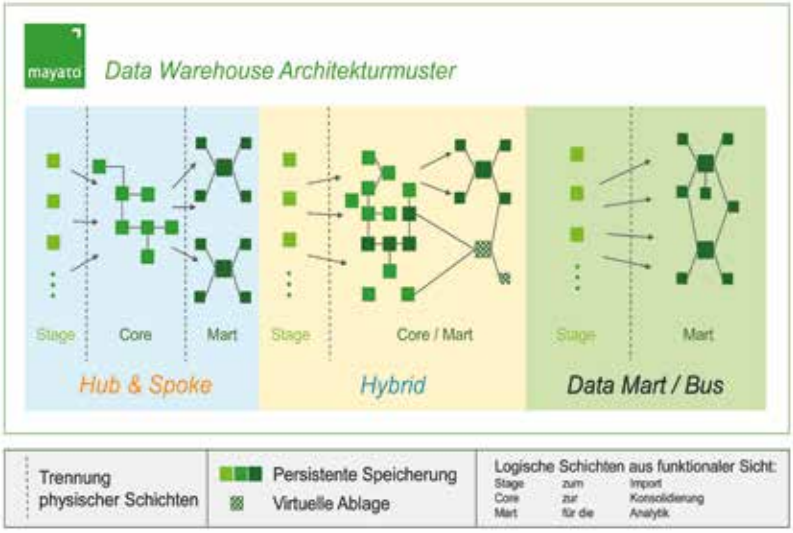


Abb. 1: Schichtenkomposition von Standard-DWH-Architekturen

- **Datennutzungsmuster**
  - **Zugriff** auf Query & Reporting oder Export & Download
  - **Analyse** mittels OLAP, Business Analytics, Advanced Analytics oder Data Mining
  - **Management** als Applikationsverdichtung über Dashboards, Score Cards oder Planungsanwendungen

Allein dieser an [Wel15] angelehnte Auszug an logischen Kernfunktionalitäten eines DWH definiert einen Großteil der DWH-Projekte und legt die Rahmenparameter für eine automatisierte Umsetzung fest. Im Sinne eines Baukastenprinzips bilden die Einzelteile, hier die DWH-Entwurfsmuster, die Gesamtarchitektur des DWH.

Theoretisch ist jede Kombination der Einzelteile aus dem DWH-Baukasten möglich. Ebenso sind Einzelteile gleicher Art denkbar, etwa die parallele Umsetzung einer „Hub & Spoke"- und „Data Lake"-Architektur zur separaten Speicherung von strukturierten und unstrukturierten Daten. Es gibt allerdings auch Kombinationen mit geringen Erfolgsaussichten, etwa der Einsatz von ELT auf Datenbanken, die über wenig Transformationsfunktionalität verfügen.

Durch die Festlegung der jeweiligen Entwurfsmuster wird von der technischen Ebene auf eine funktionale Ebene abstrahiert. Wie genau ein SQL-Job zur Ausführung der einfachen Versionierung technisch umgesetzt wird, bestimmt das Entwurfsmuster. Notwendig für die Umsetzung sind die beschreibenden Metadaten, über welche die fachlichen Attribute entschieden werden. Gleichzeitig wird darüber festgelegt, ob eine neue Version des Datensatzes angelegt werden soll bzw. was die korrespondierenden Schlüssel sind, was wiederum aus dem Schlüsselkonzept resultiert. Die technische Umsetzung erfolgt dann automatisiert durch die DWA-Lösung: Es muss im klassischen Sinn nichts mehr programmiert werden.

Mit DWA zum DWH nach dem Baukastenprinzip

Am Beispiel einer generischen DWA-Lösung wird eine Applikation skizziert, die metadatengetrieben auf Basis von Data Vault sämtliche Fortschreibungsprozesse über die Datenschichten dynamisch generiert. Die benötigten Definitionen werden dabei über Metadaten verwaltet, von der Analyse der Quelldaten bis zur automatischen Erstellung technischer Prozesse auf Datenbankebene. Diese Verzahnung der Metadaten über alle Schichten hinweg und die konsequente Nutzung von Entwurfsmustern ermöglichen eine vollständige Automatisierung.

Die generische DWA-Lösung geht dabei im Vergleich zu klassischen DWA-Werkzeugen noch einen Schritt weiter: Sie legt Datenfortschreibungsprozesse nicht nach deren Definition persistent ab, sondern erzeugt diese dynamisch während der Ausführung eines Datenladeprozesses und führt sie unmittelbar auf der Datenbank aus. Dadurch findet sich keine Zeile Transformationscode auf der Datenbank, sodass Entwickler keinen Code beeinflussen können, da alle Ausführungsobjekte nur temporär und über Metadaten gesteuert existieren. Für die Nachvollziehbarkeit protokolliert das System jeden Ausführungsschritt.

Den Vorteilen einer sehr starken Standardisierung mit minimalem Code stehen aber auch Nachteile gegenüber: Aus Performance-Gründen ist der Einsatz einer In-Memory-Datenbank notwendig. Fachliche Transformationen beschränken sich auf „Bordmittel" der Datenbank und damit auf SQL. Optimierungen sind nur allgemein gültig umsetzbar, nicht im Nachhinein und nicht situationsabhängig. Nachträgliche Änderungen am dynamischen Generierungsprozess wirken sich auch auf alle bestehenden Objekte aus und sind dementsprechend umfassend und wiederholend zu testen.

Eine generische DWA-Lösung zielt darauf ab, sämtliche DWH-Artefakte in einer Instanz, der zentralen Datenbank, zu bündeln. Dazu zählen sowohl die definierenden Metadaten als auch die daraus erzeugten Objekte zur Datennutzung. Für die zentrale Datenschicht bietet sich Core RAW Vault an, wie es unter anderem vom DWA-Tool Datavault Builder verwendet wird. Das nach Data Vault 2.0 [LiO15] favorisierte Konzept der natürlichen Schlüssel (Business Keys) in Verbindung mit generierten Hash-Schlüsseln wird ebenso umgesetzt wie die Möglichkeit, historische Daten in Versionen über Gültigkeitszeiträume abzulegen. Die entsprechende Umsetzung des Modells und der adäquaten Datenfortschreibung steuert die generische DWA-Lösung selbstständig, basierend auf den Metadaten.

DWA ermöglicht fachlich getriebenes Design eines DWH

Mit einer generischen DWA-Lösung können technisch affine Fachbereichsnutzer ein DWH auf Basis fachlicher Definitionen und Quelldatenstrukturen implementieren und die Agilität von DWH-Projekten fördern. Hierbei sind der Komplexität der abzubildenden Fachlichkeit jedoch Grenzen gesetzt. Diese resultieren einerseits aus der werkzeugseitig

reduzierten Transformationsfunktionalität per SQL und andererseits aus der benötigten SQL-Kenntnis des Nutzers.

Folgender Anwendungsfall demonstriert die Vorgehensweise zur Umsetzung eines einfachen DWH mit der Methodik einer generischen DWA-Lösung. Ziel ist der Aufbau eines analytischen Datenbestands für Umsatzanalysen, die nach den hierarchischen Dimensionen Verkaufskanal, Artikel und Zeit, mit parallelen Hierarchiepfaden über Woche und Monat, durchgeführt werden können.

Die Installation einer generischen DWA-Lösung erfolgt zunächst Skript-gesteuert auf der Datenbank. Über eine spezielle Dialoganwendung werden die Metadaten bearbeitet. Im ersten Schritt definiert der Nutzer auf rein fachlicher Basis die Metadaten. Im vorliegenden Beispiel werden für Umsatzanalysen drei Kennzahlen zu Menge, Umsatz und Transaktionstyp benötigt. Diese Kennzahlen sollen nach drei Dimensionen, Zeit, Verkaufskanal und Produkt auswertbar sein. Für jede Dimension werden die erforderlichen – auch parallelen – Hierarchien wie „Artikel -> Sparte -> Warengruppe“ definiert. Jede Dimensionsebene wiederum verfügt über diverse Attribute, von denen mindestens eines ein fachlicher Schlüssel (Business Key) ist.

Im zweiten Schritt werden die Datenquellen zur Befüllung des Infobedarfs festgelegt, inklusive etwaiger technischer Umsetzungsvorschriften für Attribute im Sinne einer einfachen ETL-Funktionalität. Eine generische DWA-Lösung ist in der Lage, Transformationsfunktionalitäten über SQL-Statements zu integrieren.

Mit diesen Definitionen ist das einfache Beispiel-DWH beschrieben. Alle weiteren Aufgaben und Aktivitäten übernimmt die generische DWA-Lösung im Hintergrund, basierend auf den definierten Entwurfsmustern. Sie baut anhand der Metadaten der Quelldatenstrukturen automatisiert die Datenschichten Stage, Core und Mart auf und stellt Prozesse für die Datenfortschreibung bereit, die erst beim Aufruf dynamisch generiert und ausgeführt werden.

Diese Fortschreibung kann der Anwender unmittelbar nach der Metadatendefinition auslösen (oder auch automatisch gesteuert ausführen lassen), um die Quelldaten in die Stage zu laden bzw. die Fortschreibung aus der Stage- in die Core-Schicht auszulösen. Nachdem die Daten über den Fortschreibeprozess in die Core-Schicht prozessiert wurden, stehen sie unmittelbar über die automatisch generierte View-Schicht zur Auswertung bereit.

Fazit

Aus der Vorgehensweise ist das Ziel der DWA als integrierte Entwicklungsumgebung für Modellierungs-, ETL- und Dokumentationsaufgaben sowie für das Deployment und den Betrieb/das Monitoring erkennbar. Der Einsatz eines solchen DWA-Werkzeugs macht vor allem dann Sinn, wenn eine erhöhte Volatilität in den Fachanforderungen und Unternehmensprozessen oder das Bedürfnis eines schnellen Time-to-Market besteht. Auch müssen die technischen und organisatorischen Grenzen dieser Vorgehensweise jeweils evaluiert werden. ETL-Werkzeuge zum Beispiel können im Bereich der Datentransformation durch mustergestützte Prozesse vielfach ersetzt werden, allerdings ist ihre Konnektivität zu anderen Systemen mittelfristig noch nicht durch DWA-Werkzeuge zu erreichen, wie in [BeR 17] beispielhaft erläutert.

Ähnlich verhält es sich mit Werkzeugen weiterer Funktionsklassen mit Berührungspunkten zu DWH-Projekten: Anforderungsanalyse, Datenmodellierung, Datenbankmanagement, ETL-Design und Implementierung, Datenqualität, Versionskontrolle, Workflow Control, User Security Management, Deployment, System Maintenance und Dokumentation. Die wesentliche Herausforderung für DWA-Tools ist der schmale Grat zwischen optimierter Individualität und generischer Flexibilität.

Ging man in DWH-Projekten lange Zeit den Weg externer, hochspezialisierter Systemkomponenten mit ETL- und Analysefunktionalitäten, so konzentriert sich DWA allgemein und die beispielhaft vorgestellte generische DWA-Lösung im Speziellen wieder auf die zentrale Datenbankkomponente mit dem Fokus der zentralen und systeminternen Datenspeicherung und -verarbeitung. Dies begründet auch die Fokussierung vieler DWA-Werkzeuge auf eine dezidierte DWH-Datenbank, vielfach Microsoft SQL Server; darunter fallen DWA-Werkzeuge wie biGENiUs [Tri 17] und TIMEXTENDER [Tim 15]. Sie setzen das automatisch um, was sich über eine lange Zeit mit individuellen Entwicklungsvorlieben standardisiert hat. Einen interessanten Ansatz liefern In-Memory-Datenbanken, die die weitere Entwicklung von DWA-Tools, auch im Bereich generischer Prozesse, stark beeinflussen werden.

Quellen

[BeR17] Beles, P. / Raetz, U.: Data Vault als Grundlage für DevOps – Data Warehouse Automation. Online-Themenspecial Data Warehouse Automation, 2017, http://www.sigs.de/publications/bi/2017/DataWarehouseAutomation/raetz_beles_BIS_OTS_DWA_2017.pdf, abgerufen am 30.6.2017

[Eck15] Eckerson, W.: Data Warehouse Automation Tools – Product Categories and Positioning. Eckerson Group 2015

[Li015] Linstedt, D. / Olschimke, M.: Buildung a Scalable Data Warehouse with Data Vault 2.0. Morgan Kaufmann 2015

[Tim15] TimeXtender Software: Data Warehouse Automation. 2015, www.timextender.com, abgerufen am 30.6.2017

[Tri17] TRIVADIS AG: What is Data Warehouse Automation? 2017, www.bigenius.info, abgerufen am 30.6.2017

[Wel15] Wells, D.: Data Warehouse Automation – A Decision Guide. Infocentric LLC 2015